Санкт-Петербургский государственный университет
Кафедра математического моделирования энергетических систем

Чернышов Дмитрий Олегович

Магистерская диссертация

Разработка и внедрение WFM процесса в Центр модерации и поддержки сайта Avito

Направление 01.04.02
Прикладная математика и информатика

Научный руководитель,
доктор физ.–мат. наук,
доцент
Крылатов А.Ю.

Санкт-Петербург
2019

Содержание

Введение 3
Глава 1. Описение процессов 5
1.1. Система массового обслуживания 5
1.2. WFM как управление СМО 6
1.3. Facebook Prophet 9
1.4. Генетический алгоритм 9
1.5. Apache Airflow 11
Глава 2. Задача прогнозирования входящего потока обращений . . . 14
2.1. Постановка задачи 14
2.2. Выбор модели 14
2.3. Описание модели 15
2.4. Результаты 17
Глава 3. Задача расчета оптимального количества сотрудников и их графиков работы 18
3.1. Постановка задачи 18
3.2. Выбор моделей 18
3.3. Описание моделей 19
3.4. Результаты 24
Глава 3. ETL-процесс 26
4.1. Постановка задачи 26
4.2. Описание метода 26
4.3. Результаты 27
Заключение 28
Список литературы 29

Введение

Современные технические средства позволяют разработать и автоматизировать процесс оптимального [1] планирования штата сотрудников компании с помощью методов и алгоритмов искусственного интеллекта. Для реализации необходимо формализовать процесс работы сотрудников, требования и цели.

Данная работа посвящена решению задач для центра модерации и поддержки компании Avito. На рынке уже существуют готовые WFM-решения [2] для управления CMO [3] с ожиданием, в которых имеется накопитель бесконечной емкости, но ввиду их стоимости и ограниченного функционала, было решено имплементировать собственное решение, которое можно модернизировать самостоятельно. Готовые решения зачастую не имеют открытого кода, поэтому добавление новой логики или модернизация старой производится только компанией-разработчиком.

Первая задача — прогнозирование потока обращений. Она необходима для решения второй задачи, поскольку ее решение будет являться обязательным входным параметром для обучения модели. Решить ее дает возможность библиотека fbprophet [4][5][6] от компании Facebook.

Вторая задача — расчет оптимального количества сотрудников для обработки обращений пользователей. Кроме того, необходимо оптимизировать их рабочие смены так, чтобы держать ключевые метрики и не нарушать Трудовой Кодекс РФ. Решение данной задачи возможно реализовать с помощью эвристики [7], а именно генетического алгоритма [8] с несколькими фитнес-функциями в два этапа. Первый этап — расчет необходимого количества рабочих смен для каждого дня недели. Второй этап — расчет необходимого количества сотрудников, которые могут оптимально покрыть количество рабочих смен, рассчитанное на первом этапе.

Третья задача — имплементация ETL-процесса [9] с помощью технологии Airflow [10]. Код программ, реализованный при решении первой и второй задач необходимо запускать еженедельно. Кроме того, код для каждого канала обращений должен исполняться параллельно, чтобы сократить общее время работы. Выходные данные должны записываться в хранилище в соответствующие таблицы.

Глава 1. Описение процессов

1.1. Система массового обслуживания

Обращение — запрос на обслуживание. Система массового обслуживания (СМО) [3] — система, которая производит обслуживание поступающих в нее обращений. Входящий поток обращений — совокупность заявок, поступающих в СМО. Время обработки обращения — период времени, в течение которого обслуживается обращение клиента.

Классическая СМО содержит от одного до бесконечного числа приборов. В зависимости от наличия возможности ожидания поступающими требованиями начала обслуживания СМО подразделяются на:

- Системы с потерями, в которых требования, не нашедшие в момент поступления ни одного свободного прибора, теряются.

- Системы с накопителем конечной ёмкости (ожиданием и ограничениями), в которых длина очереди не может превышать емкости накопителя. При этом требование, поступающее в переполненную СМО (отсутствуют свободные места для ожидания), теряется.

- Системы с ожиданием, в которых имеется накопитель бесконечной ёмкости для буферизации поступивших требований, при этом ожидающие требования образуют очередь. Именно данная система присутствует в нашей задаче.

Выбор требования из очереди на обслуживание производится с помощью так называемой дисциплины обслуживания. Их примерами являются:

- FCFS/FIFO (пришедший первым обслуживается первым).
- LCFS/LIFO (пришедший последним обслуживается первым).

В системах с ожиданием накопитель в общем случае может иметь сложную структуру. В нашей задаче используется FCFS/FIFO.

1.2. WFM как управление СМО

Workforce Management (WFM) [2] — методология планирования рабочего времени сотрудников компании – один из основополагающих камней в фундаменте оптимизации работы контактного центра. В качестве отдельного модуля, производящего прогнозирование нагрузки и расчет календарного планирования, Workforce Management просто бесценен для менеджеров

контактных центров. В жестких экономических условиях особенно важно быть уверенным в том, что в контактном центре работает оптимальное количество сотрудников, с соответствующими навыками и умениями, и что все они работают по наиболее продуктивному графику, который не только сводит расходы к минимуму, но и гарантирует клиентам полный доступ к контактному центру.

Главное требование к инструменту Workforce Management — быть способным точно предсказывать объем нагрузки в конкретный промежуток времени и генерировать оптимальное расписание, принимая во внимание принципы планирования местного рабочего времени и требования законодательства. Проще говоря, WFM помогает создавать, управлять и оптимизировать прогнозирование и планирование рабочих графиков, и, в результате, выводить на смену оптимальное количество сотрудников, которые эффективно справляются с обращениями клиентов.

Важность такой точности прогнозирования постоянно возрастает, так как без нее контактный центр не сможет достаточно качественно спланировать графики, отвечающие ожиданиям и требованиям бизнеса. Например, типичная ситуация в продажах: высокий уровень потерянных звонков ведет к сниже-

нию эффективности сотрудников, что, в свою очередь, может привести к снижению объемов продаж. Простым сокращением количества пропущенных звонков даже на 5% можно потенциально сэкономить миллионы.

Получить возможность всегда иметь под рукой оптимальный операторский состав — не такая простая задача. Для этого нужно быть внимательным к разнообразным деталям и внедрению определенных подпроцессов, которые выходят за рамки традиционных WFM-практик прогнозирования, составления рабочего графика и планирование отпусков, производящихся вручную. Многое из этих ежедневных видов деятельности уже знакомы большинству менеджеров контактных центров. Например, долгосрочное стратегическое планирование и бюджетирование, рекрутинг, отбор, прием на работу и тренинг персонала, электронное обучение, текущий тренинг и коучинг, рабочая активность, дисциплина и мониторинг качества, поощрения и меры по устранению недостатков, технологии речевой аналитики, сбор и анализ откликов клиентов.

Внедрение такого решения, как Workforce Management, может быстро повысить точность составления графиков, чем добиться значительно большей эффективности их работы. Это решение значительно улучшает производительность контакт-

центра — повышает занятость операторов, точность следования рабочему расписанию.

1.3. Facebook Prophet

Прогнозирование временных рядов — это весьма популярная аналитическая задача. Прогнозы используются, например, для понимания, сколько серверов понадобится сервису через год, каков будет спрос на товары в гипермаркете, или для постановки целей и оценки работы команды. Для прогнозирования временных рядов используют такие подходы, как ARIMA, ARCH и т. д. Но подбор параметров для ARIMA — сложный и трудоемкий процесс. Однако в 2017 году команда Core Data Science из Facebook [4] выпустила новую библиотеку для работы с временными рядами — Prophet.

1.4. Генетический алгоритм

Эвристический алгоритм [7] — это алгоритм решения задачи, правильность которого для всех возможных случаев не доказана, но про который известно, что он дает достаточно хорошее решение в большинстве случаев. В действительности может быть даже известно то, что эвристический алгоритм формально неверен. Его все равно можно применять, если при этом

он дает неверный результат только в отдельных, достаточно редких и хорошо выделяемых случаях, или же дает неточный, но все же приемлемый результат. Иначе говоря, эвристика — это не полностью математически обоснованный, но при этом практически полезный алгоритм. Важно понимать, что эвристика, в отличие от корректного алгоритма решения задачи, обладает следующими особенностями:

- Она не гарантирует нахождение лучшего решения.
- Она не гарантирует нахождение решения, даже если оно заведомо существует.
- Она может дать неверное решение в некоторых случаях.

Генетический алгоритм [8] — это эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путем последовательного подбора, комбинирования и вариации искомых параметров с использованием механизмов, напоминающих биологическую эволюцию. Является разновидностью эволюционных вычислений. Отличительной особенностью генетического алгоритма является акцент на использование оператора «скрещивания», который производит операцию рекомбинации решений-кандидатов, роль которой аналогична роли скрещивания в живой природе. Описание алгоритма:

- Случайным образом генерируется конечный набор пробных решений.
- Оценивается приспособленность текущего поколения.
- Выход, если выполняется критерий останова, иначе:
- Генерируется новое поколение посредством операторов селекции, скрещивания, мутации и переход к пункту 2.

В процессе селекции отбирают только несколько лучших пробных решений, остальные далее не используются. Скрещивание создает другие пары решений, элементы которой наследуются от родителей каким-либо образом. Мутация случайным образом меняет какую-нибудь компоненту пробного решения на иную.

1.5. Apache Airflow

Airflow [10] — это библиотека для разработки, планирования и мониторинга ETL-процессов [9]. Основная особенность Airflow в том, что для разработки процессов используется язык Python.

Рассмотрим основные сущности Airflow. Их суть и назначение — оптимальная организация архитектуры процессов. Основной сущностью в Airflow является это Directed Acyclic Graph

(DAG).

DAG (направленный ацикличный граф) — это некоторое смысловое объединение задач, которые необходимо выполнить в строго определенной последовательности по определенному расписанию. Airflow представляет удобный web-интерфейс для работы с графами и другими сущностями. Проектируя граф, закладывается набор вершин (операторов), с помощью которых будут построены задачи внутри графа.

Оператор — это сущность, на основании которой создаются экземпляры заданий, где описывается, что будет происходить во время исполнения экземпляра задания. Например, существуют операторы:

- BashOperator — для выполнения bash-команды.
- PythonOperator — для вызова Python-кода.
- EmailOperator — для отправки email.
- HTTPOperator — для работы с http-запросами.
- SqlOperator — для выполнения SQL-кода.
- Sensor — ожидания события. Например, нужного времени, появления необходимого файла, обновления таблицы в базе данных.

Есть и более специфические операторы:

- DockerOperator,
- HiveOperator,
- S3FileTransferOperator,
- PrestoToMysqlOperator,
- SlackOperator.

Кроме того, можно разрабатывать собственные операторы, ориентируясь на особенности, и использовать их в проекте. По сути, как только в проекте возникает часто используемый код, построенный на базовых операторах, можно задуматься о том, чтобы собрать его в новый оператор. Это упростит дальнейшую разработку и пополнит собственную библиотеку операторов в проекте. Все созданные экземпляры задач выполняет планировщик.

Планировщик задач в Airflow построен на Celery [13]. Celery — это Python-библиотека, позволяющая организовать очередь с асинхронным и распределенным исполнением задач. Со стороны Airflow все задачи делятся на пулы. Пулы создаются вручную. Обычно, их цель является ограничение нагрузки.

Глава 2. Задача прогнозирования входящего потока обращений

2.1. Постановка задачи

Необходимо построить почасовой прогноз потока обращений на год вперед для всех каналов, на которые поступают обращения. Из исходных данных есть только исторический почасовой поток обращений, сгруппированный по каналам. Допускается средняя абсолютная ошибка не более 0.1.

2.2. Выбор модели

Существует большое количество различных подходов для прогнозирования временных рядов. Поскольку активность пользователей сайта имеет сезонность, а аудитория сайта растет, можно воспользоваться тренд-сезонной моделью. Например, подойдет тройное экспоненциальное сглаживание [11] (или модель Хольта-Винтерса). Но возникает проблема с праздничным днями. Дни недели в праздники различаются, отсюда возникает неточность в прогнозе в большую сторону. Для бизнеса это лишние затраты, т.к. вышедших сотрудников будет больше, чем требуется для соблюдения бизнес-метрик. В этом случае, нам требуется модель, которая учитывает праздничные дни. Такую

возможность предоставляет библиотека fbprophet [5][6].

2.3. Описание модели

Библотека fbprophet был разработана для прогнозирования множества бизнес-показателей и делает это практически «из коробки». Также этот инструмент дает возможность гибких настроек, с помощью которых можно улучшать точность предсказательных моделей.

Модель выглядит следующим образом:

$$y(t) = g(t) + s(t) + h(t) + \epsilon(t)$$

$s(t)$ — сезонность, является главной составляющей, которая отражает за моделирование периодических изменений, связанных с недельной и годовой сезонностью. Например, недельная сезонность инициализируется через сведение категориальных признаков к вещественным, а интересующая нас годовая сезонность задается рядами Фурье.

$g(t)$ — тренд, логистическая или кусочно-линейная функция. Например, есть возможность использовать логистическую функцию, которая позволяет построить модель роста с насыщением. Это распространенное явление для сайтов или прило-

жений. Выглядит оно следующим образом – какая-либо метрика увеличивается, но при это темп этого увеличения снижается. Помимо этого, модель может определять точки изменения тренда по историческим данным. Происходит это не только автоматически, но и есть возможность задать данные точные вручную.

$h(t)$ — известные аномалии, которые нужно заранее ввести в модель. Это могут быть те самые праздничные дни.

$\epsilon(t)$ — ошибка, несет в себе информацию, не учтенную в модели.

Качество модели будем оценивать с помощью $MAPE$. $MAPE$ (mean absolute percentage error) — это средняя абсолютная ошибка.

$MAPE$ обычно используется для оценивания модели, а точнее ее качества. Кроме того, поскольку величина является относительной, используя ее, можно проводить сравнения качества на различающихся датасетах.

Еще одна полезная метрика, на которую требуется обращать внимание — это абсолютная ошибка (MAE — mean absolute error). Она необходима для того, чтобы понимать, какова ошибка прогнозирующей модели в абсолютных величинах.

2.4. Результаты

С помощью библиотеки fbprophet, удалось построить почасовые прогнозы входящего потока обращений в систему для всех заданных каналов на 1 год вперед. $MAPE$ во всех случаях не превышает значения 0.1, а время работы не превышает 30 минут. Код программы написан на языке Python 3.

Глава 3. Задача расчета оптимального количества сотрудников и их графиков работы

3.1. Постановка задачи

Задача состоит в расчете оптимального штата сотрудников для обработки обращений пользователей. Графики работы должны построены с учетом парных плавающих выходных дней. Кроме того, необходимо,чтобы выполнялся ключевой показатель — среднее время нахождения заявки в очереди в дневном разрезе [12], и была минимизирована дисперсия среднего времени нахождения заявки в очереди в почасовом разрезе для каждого дня и минизировано количество сотрудников. Используется почасовой прогноз потока обращений на неделю вперед. Также, нужно учесть ограничение на количество рабочих мест.

3.2. Выбор моделей

Поскольку формально мы имеем задачу многофакторной оптимизации, и нам необходимо уменьшить вероятность попадания в локальный экстремум, для решения будет достаточно генетических алгоритмов. В нашем случае, недостаток генетиче-

ского алгоритма только в скорости работы, но мы имеем большое ограничение в 4 часа.

3.3. Описание моделей

Алгоритм расчета необходимого количества рабочих смен

Введем в качестве особи вектор $X = (x_1, x_2, x_3, x_4)$, где ген i — это тип смены: 1) 23:00–07:00, 2) 07:00–15:00, 3) 15:00–23:00, 4) 10:00–18:00. Начальную популяцию представим в случайном виде. Оператор селекции реализуем по принципу 50/50, т. е. половина особей — «хорошие», половина — «плохие». В качестве оператора скрещивания выберем одноточечный кроссинговер, поскольку число генов мало. Скрещивание будем применять только к «хорошим» особям. Чтобы популяция не вырождалась, будем применять к «плохим» особям оператор мутации, который заменяет случайный ген случайным числом. Приспособленность будут оценивать фитнес-функции. Критерий останова — исчерпание числа поколений, отпущенных на эволюцию, которое выбрано эмпирически.

Пусть d — время, f — поток обращений, sb — очередь обращений, p — производительность сотрудника, tt — цель для среднего время ответа, bt — начальное состояние очереди обра-

щений, c — количество рабочих мест и особь X. Тогда

$$a = \begin{cases} x_1, & i \in [1, 8], \\ x_2, & i \in [9, 10], \\ x_2 + x_4, & i \in [11, 16], \\ x_3 + x_4, & i \in [17, 18], \\ x_4, & i \in [19, 24], \end{cases}$$

$$s_i = a_i p, \quad i \in [1, 24],$$

$$b_i = \begin{cases} sb + f_1 - s_1, & i = 1, \\ bi - 1 + f_i - s_i, & i \in [2, 24], \end{cases}$$

$$ba_i = \begin{cases} 0, & i \in [1, 24], \\ b_i, & i \in [1, 24], \end{cases}$$

$$sn_i = \begin{cases} s_i + b_i, & b_i \leqslant 0, \quad i \in [1, 24], \\ s_i, & b_i \geqslant 0, \quad i \in [1, 24], \end{cases}$$

$$ban_i = |ba - b_i|, \quad r_i = \frac{ba_i}{sn_i}, \quad ca_i = c - a_i, \quad i \in [1, 24].$$

Опишем фитнес-функции:

$$bn = \sum_{i=1}^{24} ban_i, \quad cn = \sum_{i=1}^{24} ca_i,$$

$$v = \frac{\sum_{i=1}^{24} \left(r_i - \frac{\sum_{j=1}^{24} r_j}{24} \right)}{23},$$

$$dv = \begin{cases} v + 1, & v \geqslant vt, \\ s_i, & v \leqslant vt, \end{cases}$$

$$la = \frac{\frac{1}{24} \sum_{i=1}^{24} ba_i}{\frac{1}{24} \sum_{i=1}^{24} sn_i + \sqrt{\frac{\sum_{j=1}^{24} \left(sn_j - \frac{\sum_{k=1}^{24} sn_k}{24} \right)^2}{23}}},$$

$$dla = \begin{cases} la + 10, & la \geqslant tt, \\ tt, & la \leqslant tt, \end{cases}$$

$$baa = \frac{b_{24}}{bt},$$

$$dbaa = \begin{cases} baa + 1000, & baa \leqslant 0{,}9, \quad baa \geqslant 1{,}1, \\ baa, & 0{,}9 < baa < 1{,}1. \end{cases}$$

Реализуем алгоритм:

1. Генерируем начальную популяцию.

2. Проводим скрещивание. Первый раз над всей популяцией, следующие разы — для «хороших» особей, так как начальная популяция в два раза меньше оперируемой в дальнейших шагах.

3. Оцениваем приспособленность. Считаем фитнес-функции для каждой особи из популяции и сортируем по значениям функций bn, cn, dv, dla, $dbaa$, v, la по возрастанию.

4. Проводим селекцию.

5. Проводим мутацию над «плохими».

6. Снова оцениваем приспособленность.

7. Снова проводим селекцию.

8. Возвращаемся к пункту 2 и зацикливаемся на заданное количество поколений.

На выходе получаем лучшую особь $X = (x_1, x_2, x_3, x_4)$. Повторяем этот алгоритм для каждого дня недели. В итоге результатом будет R (7×4)-матрица.

Алгоритм расчета необходимого количества сотрудников

Введем в качестве особи вектор $X = (x_1, ..., x_n)$, где ген x_i - агент с номером первого выходного дня (1 — означает, что выходные «Пн», «Вт» и т. д.). Количество генов (сотрудников) для каждого типа смены определим как $max(R_{ij}) * 1.33$, рассчитывая, где i — тип смены, j — день недели. Начальную популяцию представим в случайном виде. Оператор селекции реализуем по принципу 50/50, т. е. половина особей — «хорошие», половина — «плохие». В качестве оператора скрещивания выберем двухточечный кроссинговер, поскольку число генов достаточно велико. Скрещивание применяем к «хорошим» особям. Чтобы наша популяции не вырождалась (уходила из локальных экстремумов), будем использовать оператор мутации, который заменяет случайный ген случайным число. Мутацию применяем к «плохим». Критерий останова — исчерпание числа поколений, отпущенных на эволюцию (выбран эмпирически). Оценку приспособленности будем проводить по системе фитнес-функций с ограничениями. Каждую функцию мы будем оптимизировать, а точнее, устремлять к нужному значению, которое не всегда экстремум.

Общее количество сотрудников на каждый день недели для одного типа графика рассчитывается как:

$$N_i = \begin{cases} \sum_{j=1}^{n} [X_j \neq i] \& [X_j \neq i-1], & i \in [2,7] \\ \sum_{j=1}^{n} [X_j \neq 1] \& [X_j \neq 7], & i = 1 \end{cases}$$

Опишем вспомогательные функции и фитнес-функции, которые будем оптимизировать и, при необходимости, введем для них штрафы:

$$da_i = \begin{cases} N_i - M_i, & N_i - M_i \geqslant 0, i \in [1,7] \\ N_i - M_i + 100, & N_i - M_i \leqslant 0, i \in [1,7] \end{cases}$$

$$daa = \sum_{i=1}^{7} da_i$$

$$ada = \frac{dad}{\sum_{i=1}^{7} [da_i \neq 0]}$$

3.4. Результаты

С помощью двух генетических алгоритмов было рассчитано необходимое количество рабочих смен для обработки входящего потока обращений и необходимое количество сотрудников, которые смогут покрыть рабочие смены с пятидневным графиком работы и парными плавающими выходными. Кроме того,

выполняются поставленные бизнес-цели и соблюдается ограничение на количество рабочих мест. Время работы не превышает 1 часа 10 минут. Код программы написан на языке Python 3.

Глава 3. ETL-процесс

4.1. Постановка задачи

Используя технологию Airflow, необходимо имплементировать еженедельный запуск алгоритмов, удостоверившись, что данные за предыдущие периоды доступны в хранилище. Результаты работы алгоритмов записать в хранилище.

4.2. Описание метода

Для обработки данных достаточно построить направленный ацикличный граф. Корнем графа является сенсор, который проверят готовность данных. Вершины второго уровня — операторы, вызывающие работу алгоритма построения прогноза входящих обращений. Один оператор для каждого канала обращений. Вершины третьего уровня — операторы, вызывающие работу алгоритма для расчета необходимого количества рабочих смен и записывающие результаты в хранилище. Один оператор для каждого канала обращений. Вершины четвертого уровня - операторы, вызывающие работу алгоритма для расчета необходимого количества сотрудников и записывающие результаты в хранилище. Один оператор для каждого канала об-

ращений. Таким образом после готовности данных, все алгоритмы параллельно начинают работу.

4.3. Результаты

С помощью технологии Airflow был реализован процесс запуска алгоритмов. Был построен направленный ацикличный граф, который и позволяет параллельно запускать алгоритмы, проверяя, что необходимые данные доступны в хранилище.

Заключение

Для компании Avito был реализован WFM процесс, с помощью которого наиболее оптимально планируются рабочие графики для большого штата сотрудников.

На языке Python были имплементированы генетические алгоритмы. Для решения первой задачи, прогнозирования входящего потока обращений, использавалась библиотека prophet от Facebook. Решение второй задачи, поиска оптимальных рабочих смен были написаны генетические алгоритмы с несколькими фитнес-функциями. Реализована возможность изменения этих функций и их приоритетов.

Для реализации параллельного ETL-процесса был использован Apache Airflow. Для своевременного и параметризованного запуска алгоритмов на языках Python и SQL были написаны сенсор, операторы и направленный ацикличный граф.

Для визуализации результатов использовалась BI система Tableau. Возможность просматривать и обновлять этот отчет заинтересованными сотрудниками предоставляет Tableau Server.

Список литературы

1. Rai V., Chandak P. Fourier optics in nonlinear image processing Shift Planning and Scheduling For IT Service Operations Management // 9th Annual IEEE International Systems Conference SysCon. 2015.

2. Murthy G. A Scientific Tool for Workforce Management in BPO Companies, 2015.

3. Ronal A., Wilson J. Queuing Theory and Customer Satisfaction: A Review of Terminology, Trends, and Applications to Pharmacy Practice // Hospital Pharmacy. 2001.

4. Forecasting at scale [Электронный ресурс]: `URL:https://facebook.github.io/prophet/` (дата обращения: 07.05.17).

5. Taylor S., Letham B. Forecasting at scale // PeerJ Preprints, 2017.

6. GitHub - facebook/prophet: Tool for producing high quality forecasts for time series data that has multiple seasonality with linear or non-linear growth. [Электронный ресурс]: `URL:https://github.com/facebook/prophet/` (дата обращения: 07.05.17).

7. Mijwel M. Heuristic Algorithms, 2015.

8. Shrestha A., Mahmood A. Improving Genetic Algorithm with Fine-Tuned Crossover and Scaled Architecture // Journal of Mathematics. 2016. Vol. 2016.

9. Köppen V., Brüggemann B., Berendt B. Designing Data Integration: The ETL Pattern Approach // Cepis Upgrade. 2011. Vol. 13. P. 49–55.

10. Apache Airflow Documentation [Электронный ресурс]: URL: https://airflow.apache.org/project.html/ (дата обращения: 28.11.17).

11. Chatfield C. The Holt-Winters Forecasting Procedure // Journal of the Royal Statistical Society. Series C (Applied Statistics). 1978. Vol. 27. №˙ 3, P. 264–279

12. Чернышов Д. О., Паршин А. А., Семке А. А., Голубев Р. И. Оценка среднего времени нахождения заявки в многоканальной системе массового обслуживания с неограниченной очередью // Аллея науки. 2017. Т. 2. no:11. С. 328–331.

13. Celery — Distributed Task Queue [Электронный ресурс]: URL: http://docs.celeryproject.org/en/latest/ (дата обращения: 28.11.17).